REMARKS/ARGUMENTS

These remarks are submitted responsive to the Office Action of July 28, 2004 (Office Action). As this response is timely filed within the 3-month shortened statutory period, no fee is believed due.

In paragraphs 4-5, the Examiner objected to claims 1-6 and 16-21 because of minor informalities, which have been corrected. In light of these corrections, shown in the amendments to the claim section, Applicants respectfully request the claim objections be withdrawn.

In paragraphs 6-7, the Examiner has rejected claims 11 and 13 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Publication 2003/0069908 to Anthony, *et al.* (Anthony). In paragraphs 8-9, the Examiner has rejected claims 1-3, 5-6, 16-18, and 20-21 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of U.S. Patent Publication 2002/0016828, to Daugherty, *et al.* (Daugherty). In paragraph 10, the Examiner has rejected claims 4 and 19 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Daugherty, in further view of "Understanding UML The Developer's Guide with a Web-Based Application in Java", Harmon, *et al.*, Morgan Kaufmann Publishers, Inc. 1998, pp. 214-253 (Harmon). In paragraph 11, the Examiner rejected claims 7-8 and 10 under Anthony in view of "Laura Lemay's Web Workshop JavaScript", Lemay, *et al.*, Sams, 1996, pp. 7-9 (Lemay). In paragraph 12, the Examiner rejected claims 9 and 14-15 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Lemay and in further view of Harmon. In paragraph 13, the

Examiner rejected claim 12 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Harmon.

In response to the Office Action, the Applicants have amended claim 13 to make claim 13 dependent upon claim 12 as opposed to claim 11. Additionally, Applicants have amended claims 1 and 16 to delete the reference to "generating and saving a footer for the selected markup language". No new matter has been added.

Prior to addressing the rejections on the art, a brief review of the Applicants' invention is in order. Applicants' claimed subject matter concerns a state chart modeling and processing system in which a state chart diagram can be produced in a state machine modeling tool, a markup language representation can be defined for the state chart diagram, and the markup language representation of the state chart diagram can be executed by a separate state machine run-time engine. State chart diagrams represent a behavioral view of a system.

Turning to the rejections of the art, claims 11 and 13 have been rejected under 35 U.S.C. § 102(e) as being anticipated Anthony. Anthony discloses a technique to translate structural diagrams (such as class diagrams in UML) to XML and vice-versa. A structural diagram describes the static or structural parts of the system and the relationships between them. A structural diagram can contain classes, relationships among classes, and cardinality for each relationship as well as attributes and services for each class.

For example, as shown in FIG. 1 of Anthony, footwear is one subclass of a clothing class, and sandals is a subclass of footwear. Footwear can inherit all attributes and method of the clothing class and can additionally have footwear specific attributes, methods, and parameters. Similarly, sandals can inherit all attributes and method of the footwear class and can additionally have sandal specific attributes, methods, and parameters.

In contrast to a structural chart or diagram, a state chart specifies the behavior of an object and how that behavior differs from state to state. Specifically, state chart diagrams capture the life cycles of objects, subsystems, and systems. State chart diagrams indicate what states an object can have and how different events affect those states over time. An object transitions from one state to another when an event occurs.

For example, states for a "footwear" object may include in-stock, out-of-stock, on-order, etc. A transition affecting the state for the "footwear" object can include events, conditions, and actions. For instance, when a customer buys a pair of a certain type of footwear (such as sandals) an inventory counter for sandals can be reduced by one. When the inventory counter reaches zero, an "out of stock" event can occur that causes sandals to move from an in-stock state to an out-of-stock state. The transition can cause one or more actions to execute, such as placing an order for additional pairs of sandals. As soon as an ordering system confirms an order sandals event, the state of sandals can change from an out-of-stock state to an on-order state.

Applicants note that Anthony is silent in regard to state chart diagrams or to modeling the behavior or lifecycle of a software system. State chart diagrams are as different from structural diagrams detailed by Anthony as they are from other software models, such as an entity relationship diagram (ERD) which models data for database tables. To one of ordinary skill in the art, techniques utilized with structural diagrams (such as class diagrams) are fundamentally different than those utilized with entity relationship diagrams (to model data) and state chart diagrams (to model behavior). Teachings specific to one type of software model are generally considered to be non-applicable to teachings specific to another type of software model. In this sense, the teachings of Anthony (directed towards structural diagrams) are in a non-analogous field from Applicant's teachings (directed towards state diagrams), which shall become increasingly apparent throughout this reply.

Referring to claim 11, Applicants claim:

> A system for linking a state machine modeling tool with a state machine run-time engine comprising:
>
> a state chart diagram generated by the state machine modeling tool, said state chart diagram comprising state chart data, said state chart data comprising state chart names, transition data and composite state actions; a state action parser for parsing said composite state actions into component state actions; and, a markup language formatter for formatting said state chart data and component state actions according to a selected markup language.

The translator of Anthony is a structure-modeling tool (structural) that is different from the Applicants claimed state machine modeling tool (behavioral or process flow). Applicants note that the **UML model utilizes separate models** (including a UML state machine diagrams and UML class diagrams), which is indicative of the fact that the model of **Anthony is a different type of software model** (structural) that can be used in conjunction with the Applicants' claimed state machine model (behavioral) but cannot be used interchangeably with the Applicants' claimed model. Anthony details a different model, having a different structure, different mechanisms, and used for a different purpose. Anthony's model is not analogous to the Applicants' model.

In paragraph 7, class names in squares and class instances in circles from Anthony are held by the Examiner to be the same as state chart names and transition data. Respectfully, class names are different from state chart names and class instances are fundamentally different from transition data. Non-analogous items are being compared (i.e. apples are being called oranges).

By definition, a class is the most general user defined type of an object-oriented language that defines variables for holding object data and methods or functions for caring out programmatic actions against objects. Instances are objects of a particular class or data members of an object class. For example, a Class object could be Tennis Shoe having variables for size, brand, cost, and the like. Class methods could include

"put shoe on foot", "tie shoe", "take shoe off foot." A class instance can be Paula's Right Shoe.

By definition, a state chart names represent one or more processing flow states that an object can have. In a shoe store example, state chart names can include names like in-stock, out-of-stock, on-order, and shipped, where each state chart name represent processing flow states pertaining to tennis shoes. Transition data can include events, conditions, and actions associated with a state chart name. Transitions can cause a particular event to fire when the processing flow state changes and/or can cause an action to initiate upon a state change. Conditions can be imposed on both actions and events further refining when the events and actions occur.

Further in paragraph 7, the Examiner has likened parsing arrows into parsing composite state actions. The two are dissimilar. Dashed lines in Anthony indicate a class/subclass relationship between the concepts represented by the vertices of the graph. The solid lines in Anthony indicate IS-A relationships (sandals is a footwearing, for example). The graphs in Anthony fail to contemplate any processing flow state (behavior) and instead indicate inheritance (structure).

Referring to claim 13, the Examiner asserts that the translator of Anthony corresponds to the state machine modeling tool of the present invention. In addition to the obvious difference that the translator is for structural models and not state machine models, the translator translates an XML document having one DTD to a different XML

document with a different DTD. In contrast, the Applicants convert a UML representation (not written in XML) to an XML document, where the XML document represents a UML-compliant state machine model.

In light of the above, Anthony fails to teach claimed limitations of the Applicants invention. Anthony is silent with regard to a state machine-modeling tool of any ilk, which Applicants expressly claim. More specifically, (1) class names in circles or instances of classes in circles are not analogous or even related to state chart names and transition information, (2) solid and dotted arrow lines in Anthony are not analogous to or even related to state actions or composite state actions, and (3) Anthony teaches translating one XML document to another XML document (with a different DTD) and does not teach or contemplate translating a UML representation to an XML document. Accordingly, the § 102(e) rejections with regards to claim 11 and 13 should be withdrawn, which action is respectfully requested.

In paragraphs 8-9, the Examiner has rejected claims 1-3, 5-6, 16-18, and 20-21 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Daugherty. Daugherty teaches a Web page rendering architecture, which fails to cure the deficiencies of Anthony. The deficiencies of Anthony being that Anthony fails to contemplate a state machine model (for modeling process flows, object lifecycle, or object "behavior") and instead teaches a structural model.

Referring to claims 1 and 16, Applicants claim the steps of:

loading **state chart data** corresponding to a **state chart diagram** through an interface to a **state machine modeling tool**;

generating header data in accordance with a selected markup language;

for each **state** specified in said state chart data, retrieving a state name and **state transition data** from said state chart data;

formatting said retrieved state names and corresponding **state transition data** according to said selected markup language; and,

saving said header data, and said formatted state names and state transition data in a document formatted according to said selected markup language.

Applicants note that Anthony does use extremely general terminology which at first glance may seem similar to the terminology of the present application, but in the context of Anthony is limited to structural diagrams and models (class diagrams), which is why the Applicants believe that the Examiner mistakenly has equated disparate concepts disclosed in Anthony with concepts of the present invention. For example, in paragraph 0042, a graph is defined in extremely general terms initially, but the relationship of the graph is defined in terms of a class and subclass ("In the top-down relationship, the concepts are related as class and subclass") making it evident that the graph is a structural or class-based graph. Also models are generically defined, as are complex models and model types in paragraph 0047. However, as noted in paragraph 0055 and 0056, a "general user definable model type" has a subclass facet specifier that only makes sense if the model is a class diagram or similar structural model.

Consequently, despite the generalized terminology used in Anthony, the only model type that Anthony provides any teachings for is a class model or structural model.

Consequently, Anthony fails to contemplate performing operations against **state chart data**, **a state chart diagram**, **a state machine modeling tool**, or **state transition data.** Further, translating a structural diagram from XML with one DTD to another structural diagram with another DTD fails to teach or suggest the loading, generating, retrieving, formatting, and saving steps claimed by the Applicants, where each step is specifically claimed to generate a document (a state diagram) formatted for a selected markup language from **state data** automatically extracted from a **state machine modeling tool**.

In paragraph 10, the Examiner has rejected claims 4 and 19 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Daugherty, in further view of Harmon. In paragraph 11, the Examiner rejected claims 7-8 and 10 under Anthony in view of Lemay. In paragraph 12, the Examiner rejected claims 9 and 14-15 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Lemay in further view of Harmon. In paragraph 13, the Examiner rejected claim 12 under 35 U.S.C. § 103(a) as being unpatentable over Anthony in view of Harmon.

Anthony, Daughty, Harmon, Lemay, and combinations thereof, each fails to teach or suggest performing operations against state chart data, a state chart data, a state machine modeling tool, or state transition data. None of Anthony, Daughty, Harmon,

Lemay, and combinations thereof mention or contemplate state machine modeling tools or state charts. Accordingly, Anthony, Daughty, Harmon, Lemay, and combinations thereof fail to teach or suggest the Applicants' claimed invention.

Referring specifically to claim 2, the XML translation of Anthony does not teach or suggest anything relating to composite state actions or individual state actions.

Referring to claims 4 and 9, Applicants respectfully note that claims 4 and 19 explicitly perform a translation from a UML model in a modeling tool to an XML representation. As noted in the Applicants' specification from line 29 of page 1 to line 17 of page 2, the UML representation is typically NOT an XML representation, but is instead a modeling tool specific representation. Anthony teaches only translating from an XML document to a different XML document. Harmon provides a basic overview of UML. Other than impermissible hindsight based on the Applicants' own claimed invention, no reference is made to converting a UML representation of a state machine modeling tool to XML by either Anthony, Daugherty, Harmon, or any combination thereof.

Referring to claim 7, Applicants claim an add-in script to a state machine modeling tool for formatting state chart data into a markup language representation according to a second markup language. This claim is novel based upon the logic inherent in the script for accessing state chart data from a UML state machine model that is not taught or suggested by Anthony, Lemay, nor any combination thereof. Lemay

teaches the combining of a Javascript program with HTML to perform certain functions. Lemay does not teach or suggest that one of these "functions" is to access state chart data from a UML state machine model. Further, HTML is different in concept and principle from UML. One of ordinary skill in the art would need to apply the logic taught by the Applicants (and not contemplated by Lemay or Anthony) to complete the transformation from UML to a corresponding XML representation for state machines. Accordingly, the references fail to teach or suggest the Applicants claimed invention.

In summary, a fundamental difference exists between state diagrams (that can model the lifecycle changes of a software object, and events triggered at various lifecycle states responsive to conditions of the software object for each state) and structural diagrams (that model the architecture of a software object). Different software modeling techniques, tools, standards, and methodologies are used for modeling state diagrams and structural diagrams.

Each diagram type is used by different artisans at different stages in the software development process for different purposes. Concepts applicable to structural diagrams (such as inheritance) and concepts applicable to state diagrams (such as state transitions) cannot be ported from one diagram type to another. Applicants teach a system and method that rely upon the structure, concepts, and principles of state diagrams. None of the cited references provide teachings regarding state diagrams and are instead directed towards structural diagrams. Consequently, for purposes of the claimed invention,

teachings pertaining to structural diagrams (as provided by the cited references) are irrelevant and provide no teachings or suggestions applicable to the claimed invention. Even should the Examiner believe the references have some relevance, no teachings have been provided to bridge the gap between structural diagram teachings and state diagram teachings, a gap which would require extensive if not whole scale revisions and modifications to the teachings disclosed by Anthony and the other cited references. Hence, at very least, the claimed invention is not obvious in view of the cited references.

In light of the above, withdrawal of the 35 U.S.C. § 102 and § 103 rejections is appropriate and is respectfully requested. Applicants believe that this application is now in full condition for allowance, which action is respectfully requested. The Applicants request that the Examiner call the undersigned if clarification is needed on any matter within this Amendment, or if the Examiner believes a telephone interview would expedite the prosecution of the subject application to completion.

Respectfully submitted,

Date: 28 October 2004

Gregory A. Nelson, Registration No. 30,577
Richard A. Hinson, Registration No. 47,652
Brian K. Buchheit, Registration No. 52,667
AKERMAN SENTERFITT
Customer No. 40987
Post Office Box 3188
West Palm Beach, FL 33402-3188
Telephone: (561) 653-5000